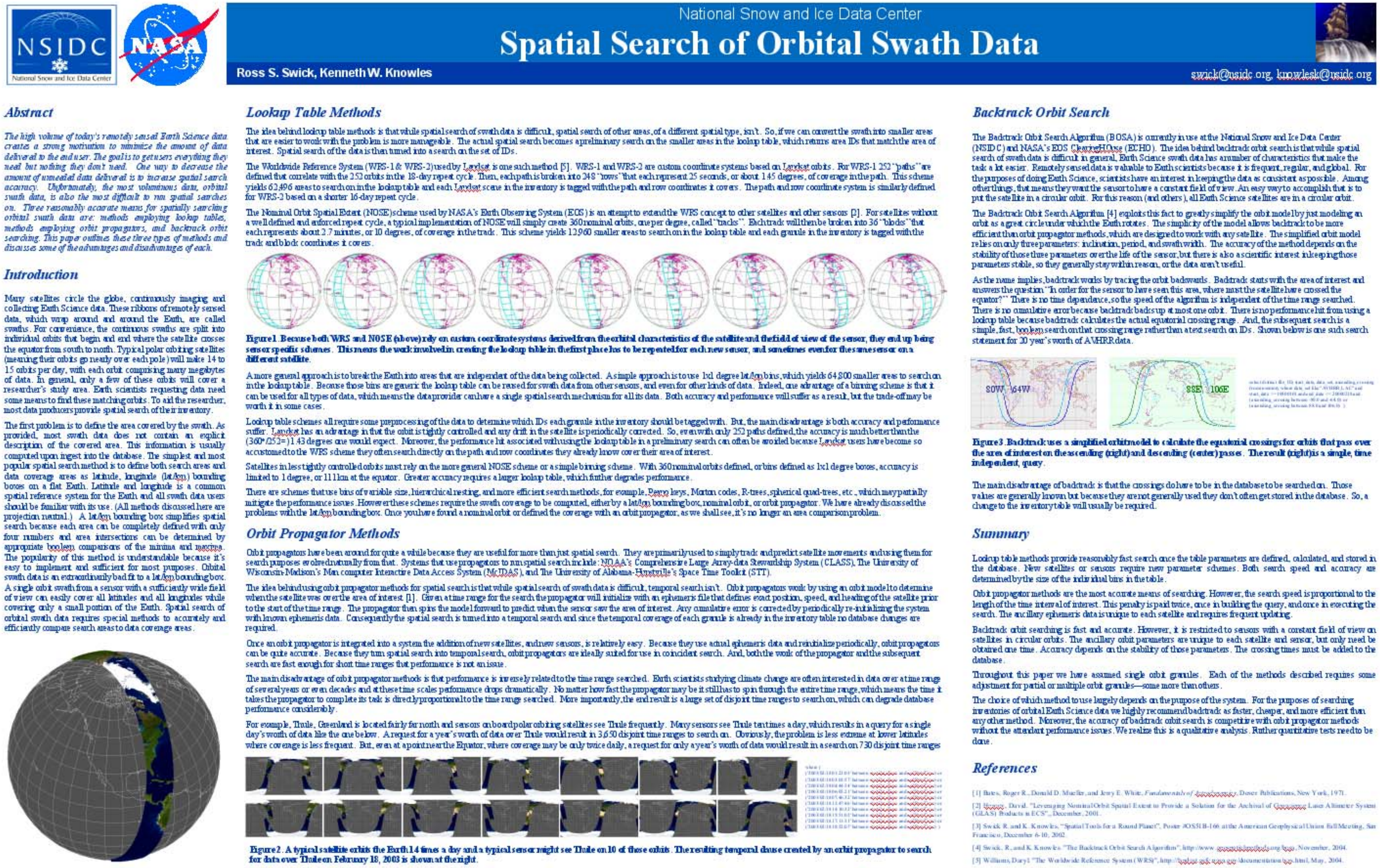
WHY DOESN'T ANYONE UNDERSTAND ANYTHING I SAY?
THREE O'CLOCK.

Spatial Search of Orbital Swath Data

Ross S. Swick, Kenneth W. Knowles
National Snow and Ice Data Center


NSIDC
National Snow and Ice Data Center
NASA
80W
64W
88E
106E

National Snow and Ice Data Center

Spatial Search of Orbital Swath Data

Ross S. Swick, Kenneth W. Knowles
swick@nsidc.org, knowlesk@nsidc.org

Abstract

The high volume of today's remotely sensed Earth Science data creates a strong motivation to minimize the amount of data delivered to the end user. The goal is to get users everything they need but nothing they don't need. One way to decrease the amount of unneeded data delivered is to increase spatial search accuracy. Unfortunately, the most voluminous data, orbital swath data, is also the most difficult to run spatial searches on. Three reasonably accurate means for spatially searching orbital swath data are: methods employing lookup tables, methods employing orbit propagators, and backtrack orbit searching. This paper outlines these three types of methods and discusses some of the advantages and disadvantages of each.

Introduction

Many satellites circle the globe, continuously imaging and collecting Earth Science data. These ribbons of remotely sensed data, which wrap around and around the Earth, are called swaths. For convenience, the continuous swaths are split into individual orbits that begin and end where the satellite crosses the equator from south to north. Typical polar orbiting satellites (meaning their orbits go nearly over each pole) will make 14 to 15 orbits per day, with each orbit comprising many megabytes of data. In general, only a few of these orbits will cover a researcher's study area. Earth scientists requesting data need some means to find these matching orbits. To aid the researcher, most data producers provide spatial search of their inventory.

The first problem is to define the area covered by the swath. As provided, most swath data does not contain an explicit description of the covered area. This information is usually computed upon ingest into the database. The simplest and most popular spatial search method is to define both search areas and data coverage areas as latitude, longitude (lat/lon) bounding boxes on a flat Earth. Latitude and longitude is a common spatial reference system for the Earth and all swath data users should be familiar with its use. (All methods discussed here are projection neutral.) A lat/lon bounding box simplifies spatial search because each area can be completely defined with only four numbers and area intersections can be determined by appropriate boolean comparisons of the minima and maxima. The popularity of this method is understandable because it's easy to implement and sufficient for most purposes. Orbital swath data is an extraordinarily bad fit to a lat/lon bounding box. A single orbit swath from a sensor with a sufficiently wide field of view can easily cover all latitudes and all longitudes while covering only a small portion of the Earth. Spatial search of orbital swath data requires special methods to accurately and efficiently compare search areas to data coverage areas.

Lookup Table Methods

The idea behind lookup table methods is that while spatial search of swath data is difficult, spatial search of other areas, of a different spatial type, isn't. So, if we can convert the swath into smaller areas that are easier to work with the problem is more manageable. The actual spatial search becomes a preliminary search on the smaller areas in the lookup table, which returns area IDs that match the area of interest. Spatial search of the data is then turned into a search on the set of IDs.

The Worldwide Reference System (WRS-1 & WRS-2) used by Landsat is one such method [5]. WRS-1 and WRS-2 are custom coordinate systems based on Landsat orbits. For WRS-1 252 "paths" are defined that correlate with the 252 orbits in the 18-day repeat cycle. Then, each path is broken into 248 "rows" that each represent 25 seconds, or about 1.45 degrees, of coverage in the path. This scheme yields 62,496 areas to search on in the lookup table and each Landsat scene in the inventory is tagged with the path and row coordinates it covers. The path and row coordinate system is similarly defined for WRS-2 based on a shorter 16-day repeat cycle.

The Nominal Orbit Spatial Extent (NOSE) scheme used by NASA's Earth Observing System (EOS) is an attempt to extend the WRS concept to other satellites and other sensors [2]. For satellites without a well defined and enforced repeat cycle, a typical implementation of NOSE will simply create 360 nominal orbits, one per degree, called "tracks". Each track will then be broken into 36 "blocks" that each represents about 2.7 minutes, or 10 degrees, of coverage in the track. This scheme yields 12,960 smaller areas to search on in the lookup table and each granule in the inventory is tagged with the track and block coordinates it covers.

Figure 1. Because both WRS and NOSE (above) rely on custom coordinate systems derived from the orbital characteristics of the satellite and the field of view of the sensor, they end up being sensor specific schemes. This means the work involved in creating the lookup table in the first place has to be repeated for each new sensor, and sometimes even for the same sensor on a different satellite.

A more general approach is to break the Earth into areas that are independent of the data being collected. A simple approach is to use 1x1 degree lat/lon bins, which yields 64,800 smaller areas to search on in the lookup table. Because those bins are generic the lookup table can be reused for swath data from other sensors, and even for other kinds of data. Indeed, one advantage of a binning scheme is that it can be used for all types of data, which means the data provider can have a single spatial search mechanism for all its data. Both accuracy and performance will suffer as a result, but the trade-off may be worth it in some cases.

Lookup table schemes all require some preprocessing of the data to determine which IDs each granule in the inventory should be tagged with. But, the main disadvantage is both accuracy and performance suffer. Landsat has an advantage in that the orbit is tightly controlled and any drift in the satellite is periodically corrected. So, even with only 252 paths defined, the accuracy is much better than the (360°/252=) 1.43 degrees one would expect. Moreover, the performance hit associated with using the lookup table in a preliminary search can often be avoided because Landsat users have become so accustomed to the WRS scheme they often search directly on the path and row coordinates they already know cover their area of interest.

Satellites in less tightly controlled orbits must rely on the more general NOSE scheme or a simple binning scheme. With 360 nominal orbits defined, or bins defined as 1x1 degree boxes, accuracy is limited to 1 degree, or 111 km at the equator. Greater accuracy requires a larger lookup table, which further degrades performance.

There are schemes that use bins of variable size, hierarchical nesting, and more efficient search methods, for example, Peano keys, Morton codes, R-trees, spherical quad-trees, etc., which may partially mitigate the performance issues. However these schemes require the swath coverage to be computed, either by a lat/lon bounding box, nominal orbit, or orbit propagator. We have already discussed the problems with the lat/lon bounding box. Once you have found a nominal orbit or defined the coverage with an orbit propagator, as we shall see, it's no longer an area comparison problem.

Orbit Propagator Methods

Orbit propagators have been around for quite a while because they are useful for more than just spatial search. They are primarily used to simply track and predict satellite movements and using them for search purposes evolved naturally from that. Systems that use propagators to run spatial search include: NOAA's Comprehensive Large Array-data Stewardship System (CLASS), The University of Wisconsin-Madison's Man computer Interactive Data Access System (McIDAS), and The University of Alabama-Huntsville's Space Time Toolkit (STT).

The idea behind using orbit propagator methods for spatial search is that while spatial search of swath data is difficult, temporal search isn't. Orbit propagators work by using an orbit model to determine when the satellite was over the area of interest [1]. Given a time range for the search the propagator will initialize with an ephemeris file that defines exact position, speed, and heading of the satellite prior to the start of the time range. The propagator then spins the model forward to predict when the sensor saw the area of interest. Any cumulative error is corrected by periodically re-initializing the system with known ephemeris data. Consequently the spatial search is turned into a temporal search and since the temporal coverage of each granule is already in the inventory table no database changes are required.

Once an orbit propagator is integrated into a system the addition of new satellites, and new sensors, is relatively easy. Because they use actual ephemeris data and reinitialize periodically, orbit propagators can be quite accurate. Because they turn spatial search into temporal search, orbit propagators are ideally suited for use in coincident search. And, both the work of the propagator and the subsequent search are fast enough for short time ranges that performance is not an issue.

The main disadvantage of orbit propagator methods is that performance is inversely related to the time range searched. Earth scientists studying climate change are often interested in data over a time range of several years or even decades and at these time scales performance drops dramatically. No matter how fast the propagator may be it still has to spin through the entire time range, which means the time it takes the propagator to complete its task is directly proportional to the time range searched. More importantly, the end result is a large set of disjoint time ranges to search on, which can degrade database performance considerably.

For example, Thule, Greenland is located fairly far north and sensors on board polar orbiting satellites see Thule frequently. Many sensors see Thule ten times a day, which results in a query for a single day's worth of data like the one below. A request for a year's worth of data over Thule would result in 3,650 disjoint time ranges to search on. Obviously, the problem is less extreme at lower latitudes where coverage is less frequent. But, even at a point near the Equator, where coverage may be only twice daily, a request for only a year's worth of data would result in a search on 730 disjoint time ranges

Figure 2. A typical satellite orbits the Earth 14 times a day and a typical sensor might see Thule on 10 of those orbits. The resulting temporal clause created by an orbit propagator to search for data over Thule on February 18, 2003 is shown at the right.

Backtrack Orbit Search

The Backtrack Orbit Search Algorithm (BOSA) is currently in use at the National Snow and Ice Data Center (NSIDC) and NASA's EOS ClearingHOuse (ECHO). The idea behind backtrack orbit search is that while spatial search of swath data is difficult in general, Earth Science swath data has a number of characteristics that make the task a lot easier. Remotely sensed data is valuable to Earth scientists because it is frequent, regular, and global. For the purposes of doing Earth Science, scientists have an interest in keeping the data as consistent as possible. Among other things, that means they want the sensor to have a constant field of view. An easy way to accomplish that is to put the satellite in a circular orbit. For this reason (and others), all Earth Science satellites are in a circular orbit.

The Backtrack Orbit Search Algorithm [4] exploits this fact to greatly simplify the orbit model by just modeling an orbit as a great circle under which the Earth rotates. The simplicity of the model allows backtrack to be more efficient than orbit propagator methods, which are designed to work with any satellite. The simplified orbit model relies on only three parameters: inclination, period, and swath width. The accuracy of the method depends on the stability of those three parameters over the life of the sensor, but there is also a scientific interest in keeping those parameters stable, so they generally stay within reason, or the data aren't useful.

As the name implies, backtrack works by tracing the orbit backwards. Backtrack starts with the area of interest and answers the question "In order for the sensor to have seen this area, where must the satellite have crossed the equator?" There is no time dependance, so the speed of the algorithm is independent of the time range searched. There is no cumulative error because backtrack backs up at most one orbit. There is no performance hit from using a lookup table because backtrack calculates the actual equatorial crossing range. And, the subsequent search is a simple, fast, boolean search on that crossing range rather than a text search on IDs. Shown below is one such search statement for 20 year's worth of AVHRR data.

Figure 3. Backtrack uses a simplified orbit model to calculate the equatorial crossings for orbits that pass over the area of interest on the ascending (right) and descending (center) passes. The result (right) is a simple, time independent, query.

The main disadvantage of backtrack is that the crossings do have to be in the database to be searched on. Those values are generally known but because they are not generally used they don't often get stored in the database. So, a change to the inventory table will usually be required.

Summary

Lookup table methods provide reasonably fast search once the table parameters are defined, calculated, and stored in the database. New satellites or sensors require new parameter schemes. Both search speed and accuracy are determined by the size of the individual bins in the table.

Orbit propagator methods are the most accurate means of searching. However, the search speed is proportional to the length of the time interval of interest. This penalty is paid twice, once in building the query, and once in executing the search. The ancillary ephemeris data is unique to each satellite and requires frequent updating.

Backtrack orbit searching is fast and accurate. However, it is restricted to sensors with a constant field of view on satellites in circular orbits. The ancillary orbit parameters are unique to each satellite and sensor, but only need be obtained one time. Accuracy depends on the stability of those parameters. The crossing times must be added to the database.

Throughout this paper we have assumed single orbit granules. Each of the methods described requires some adjustment for partial or multiple orbit granules—some more than others.

The choice of which method to use largely depends on the purpose of the system. For the purposes of searching inventories of orbital Earth Science data we highly recommend backtrack as faster, cheaper, and more efficient than any other method. Moreover, the accuracy of backtrack orbit search is competitive with orbit propagator methods without the attendant performance issues. We realize this is a qualitative analysis. Further quantitative tests need to be done.

References

[1] Bates, Roger R., Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics*, Dover Publications, New York, 1971.

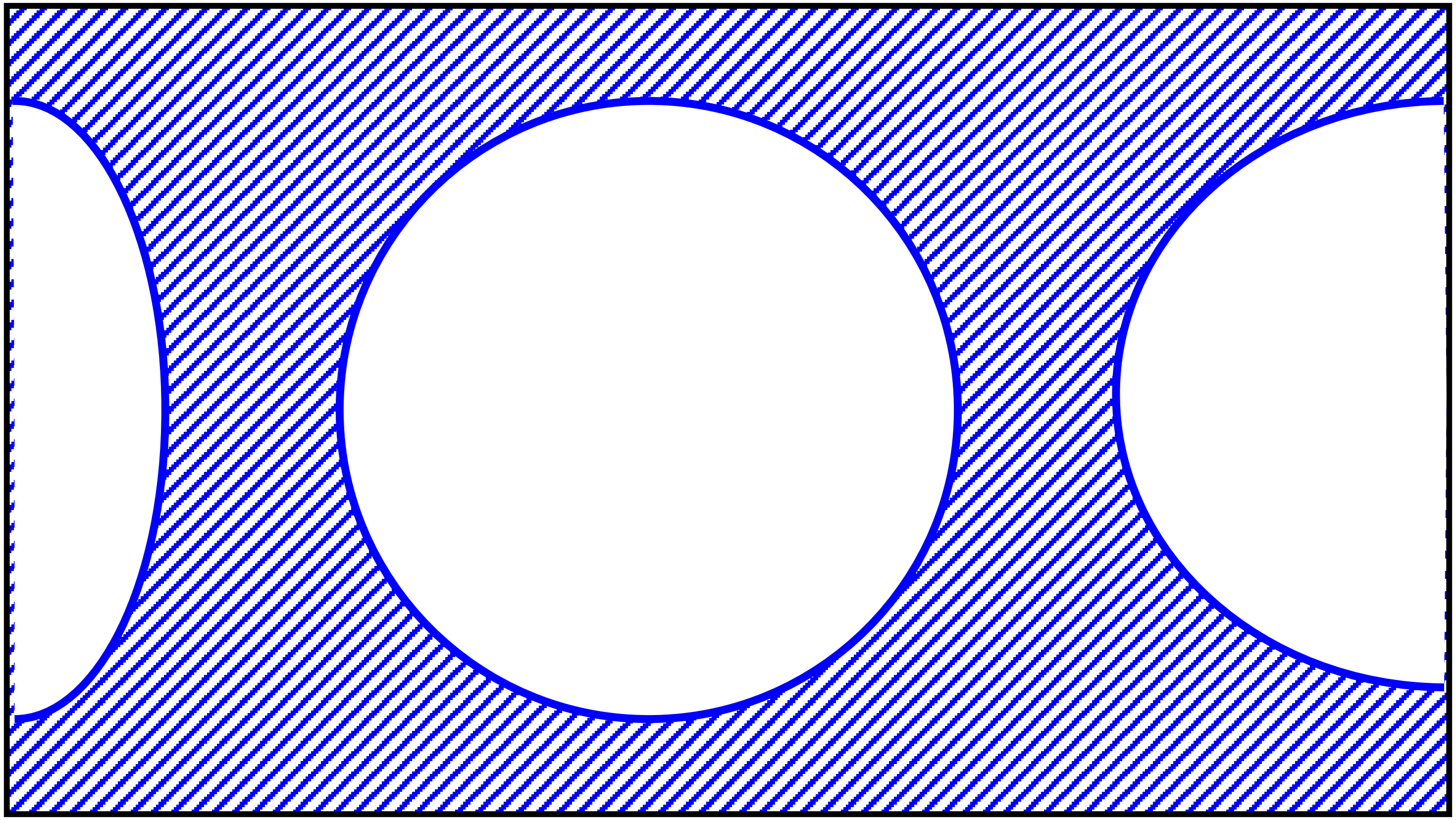
[2] [illegible], David. "Leveraging Nominal Orbit Spatial Extent to Provide a Solution for the Archival of Geoscience Laser Altimeter System (GLAS) Products in ECS", December, 2001.

[3] Swick R. and K. Knowles, "Spatial Tools for a Round Planet", Poster #OS51B-166 at the American Geophysical Union Fall Meeting, San Francisco, December 6-10, 2002.

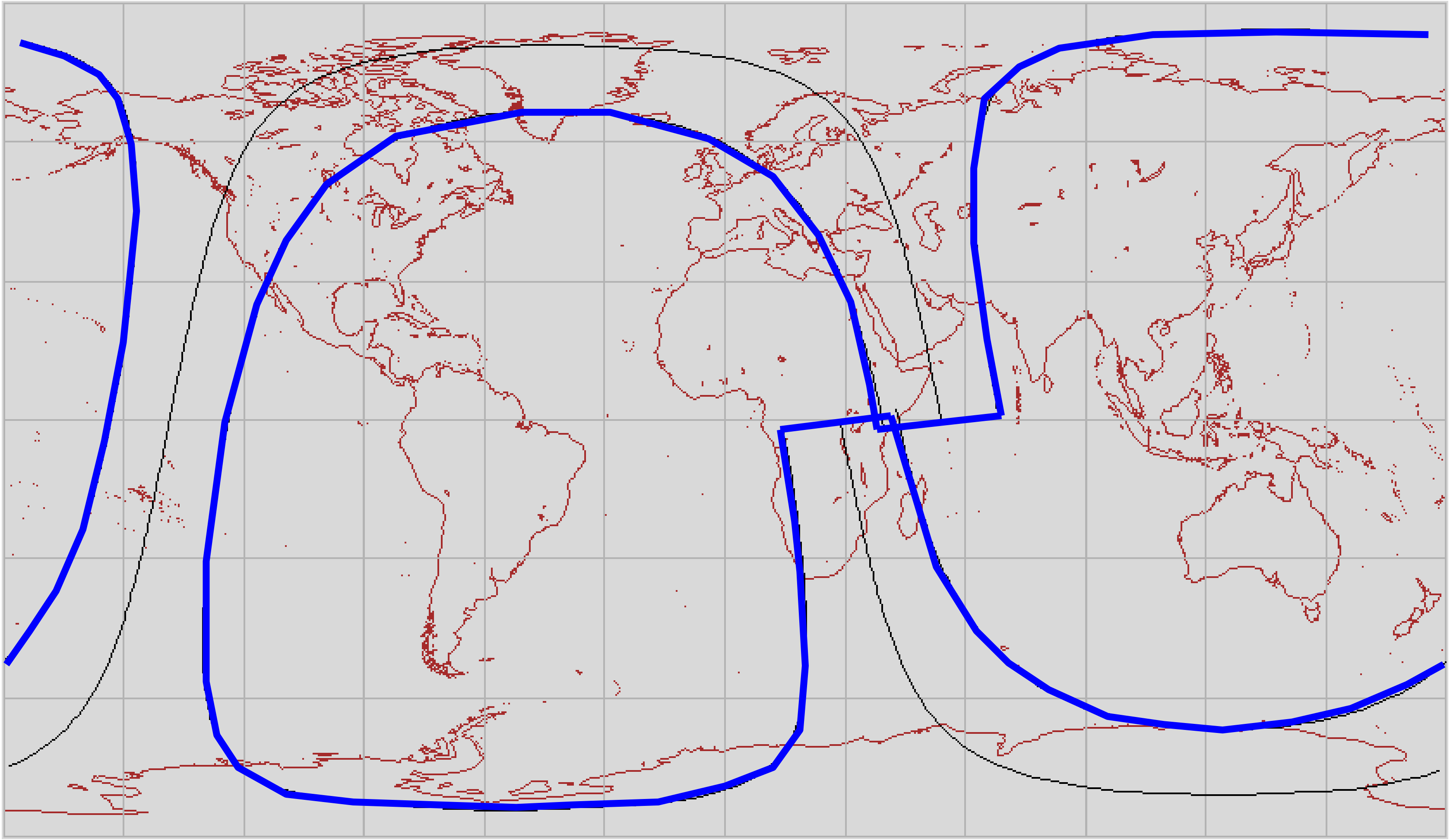
[4] Swick, R., and K. Knowles. "The Backtrack Orbit Search Algorithm", http://www.geospatialmethods.org/bosa, November, 2004.

[5] Williams, Daryl "The Worldwide Reference System (WRS)", http://landsat.gsfc.nasa.gov/documentation/wrs.html, May, 2004.

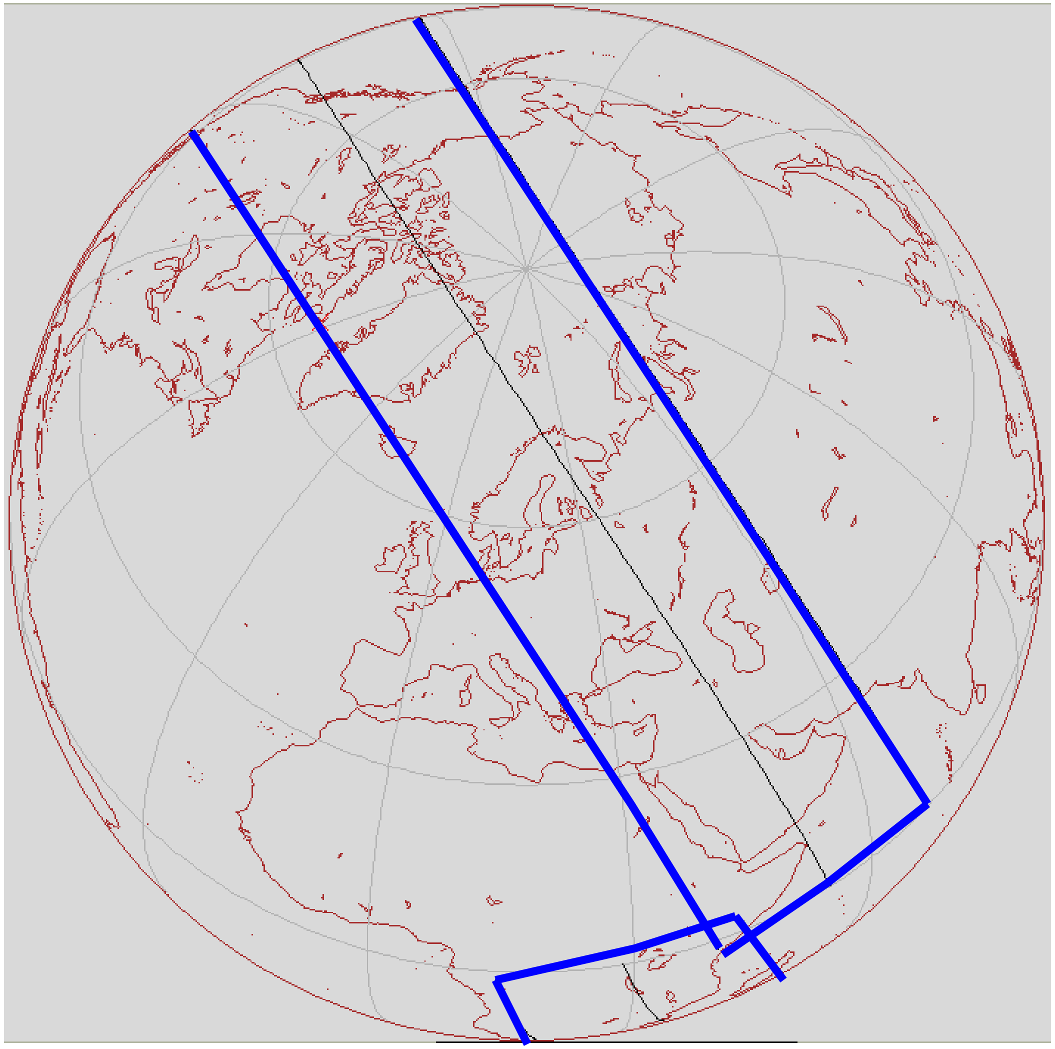
Weird Shape



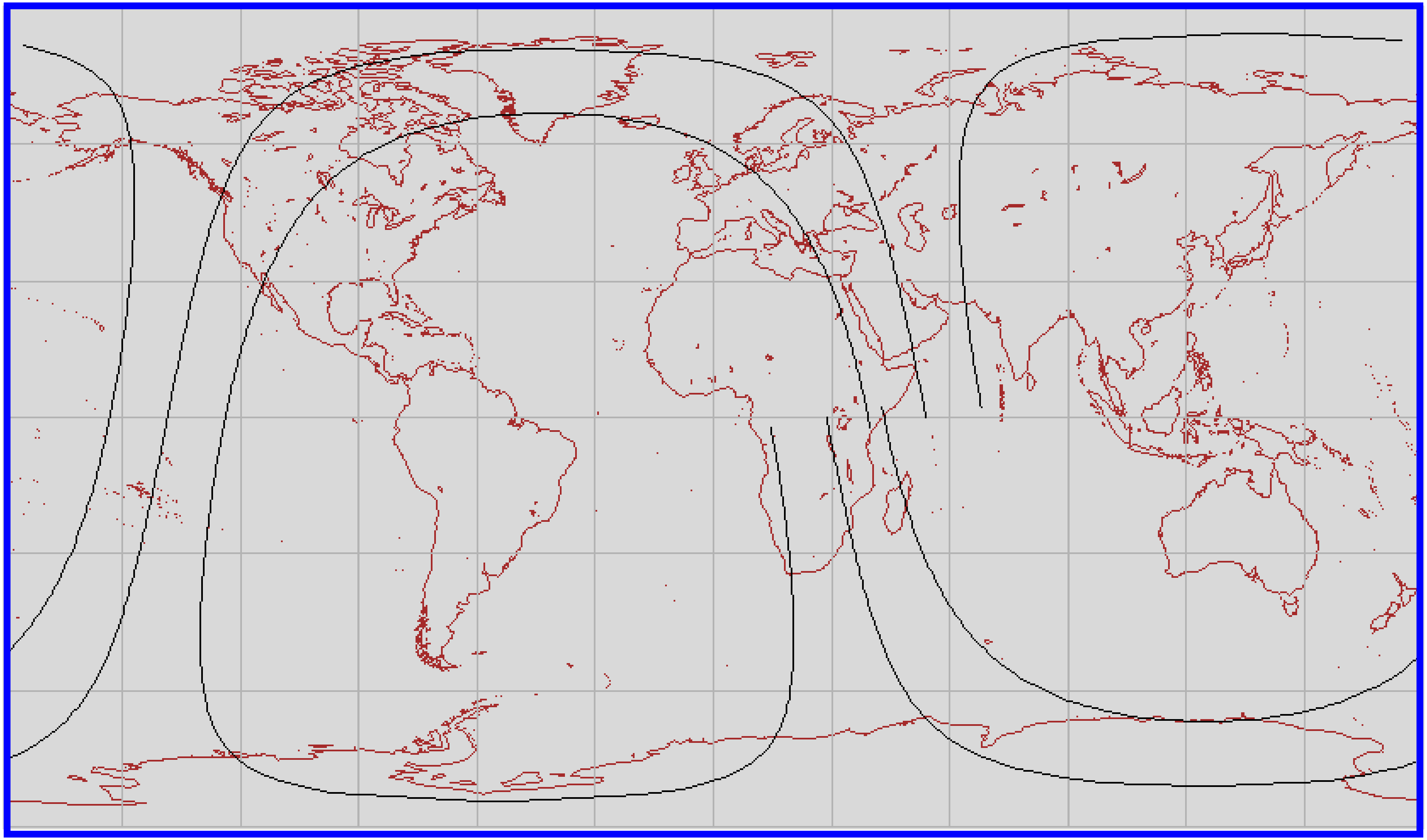
Full Orbit - Coverage



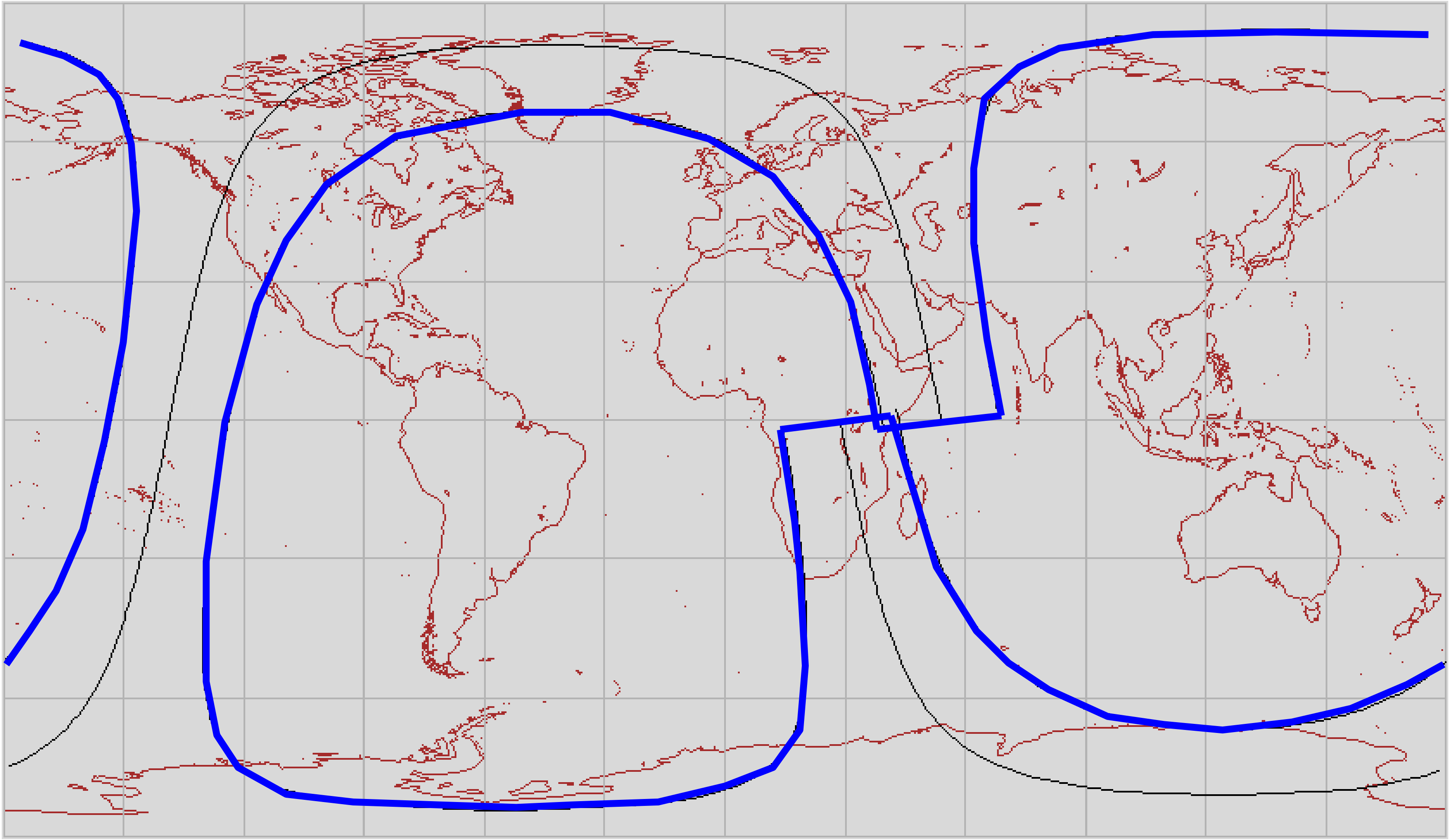
Full Orbit - Coverage



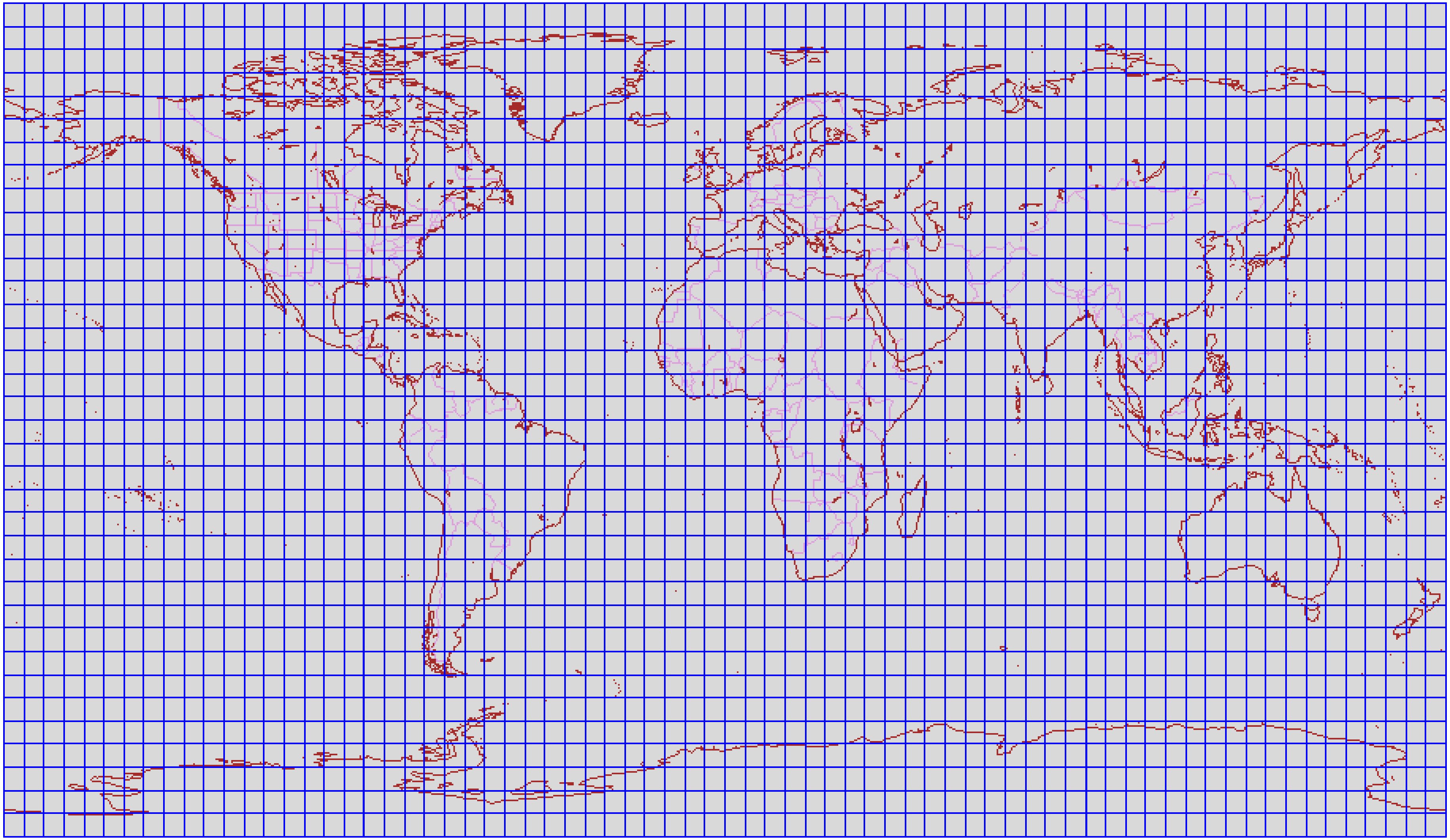
Full Orbit - LLBox



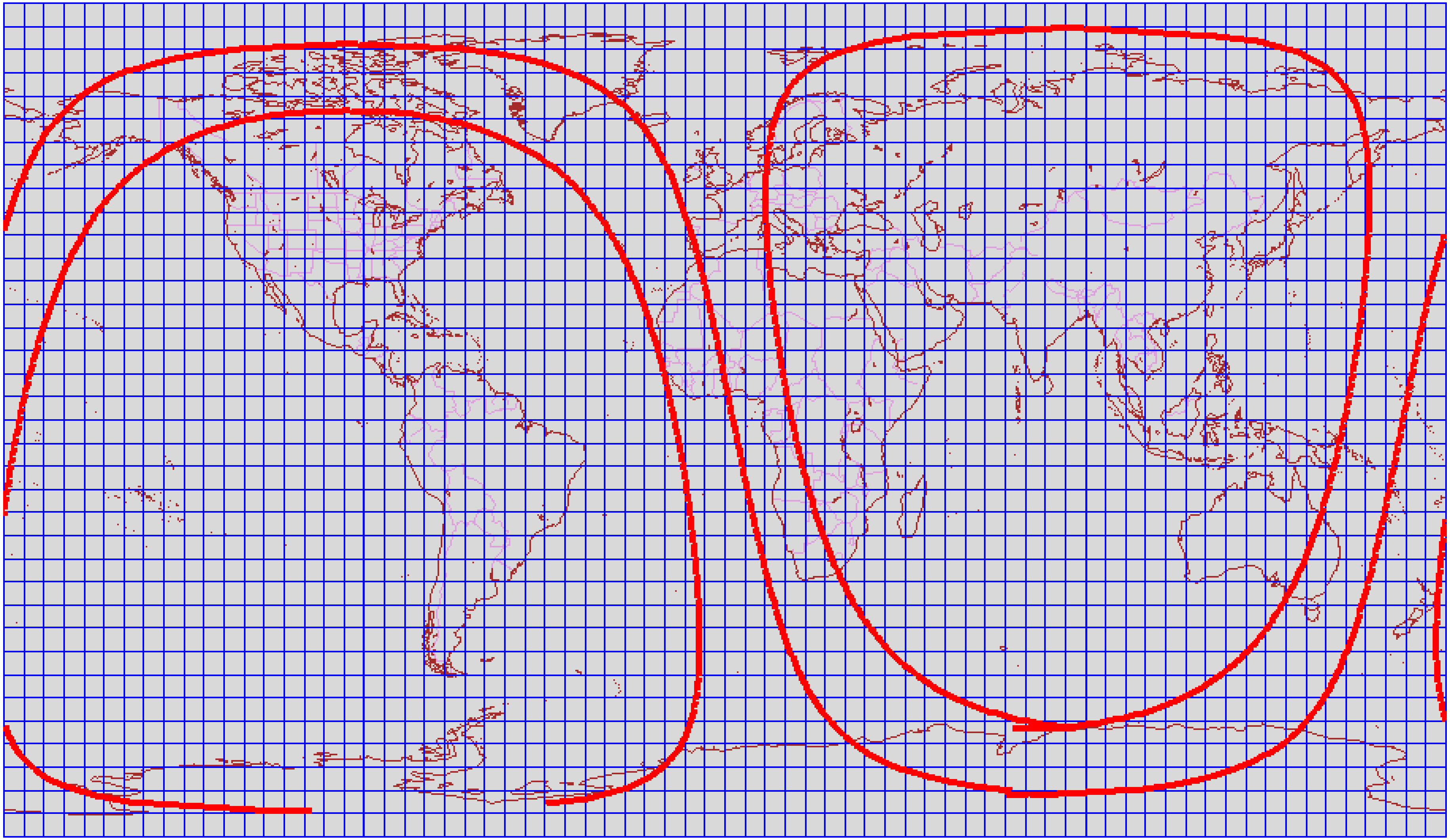
Full Orbit - Coverage



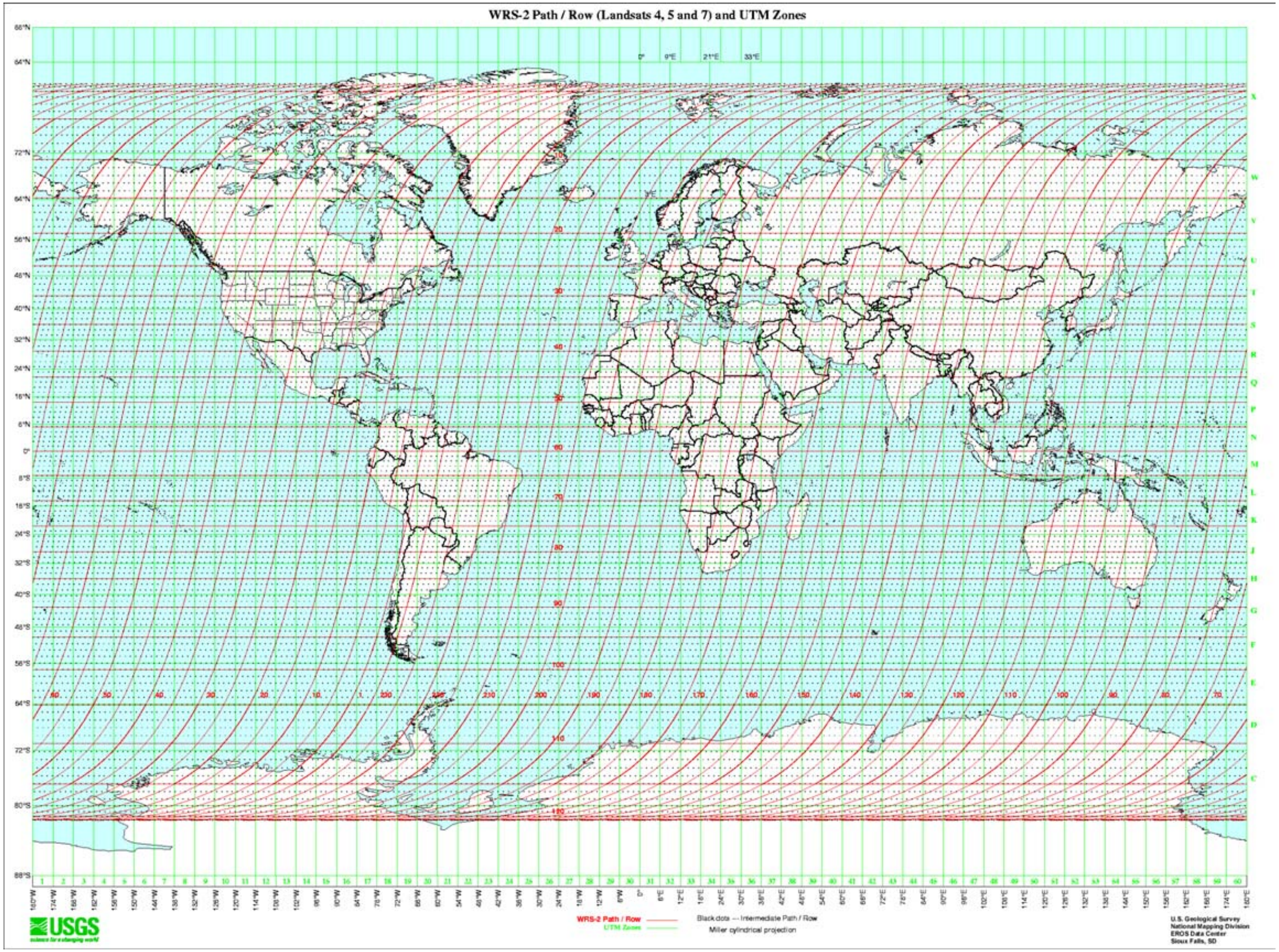
Binning/Lookup Tables



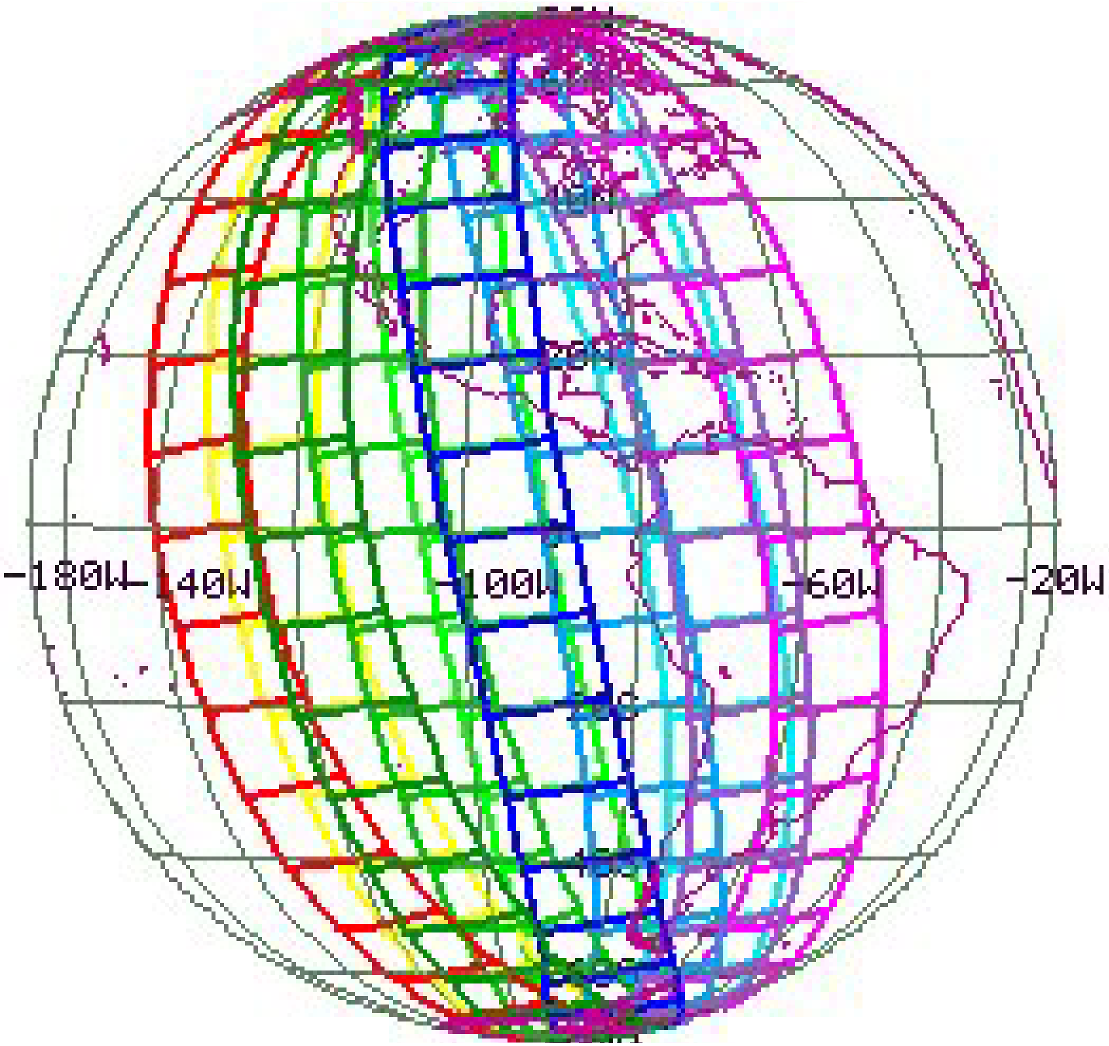
Full Orbit – Lat/Lon bins



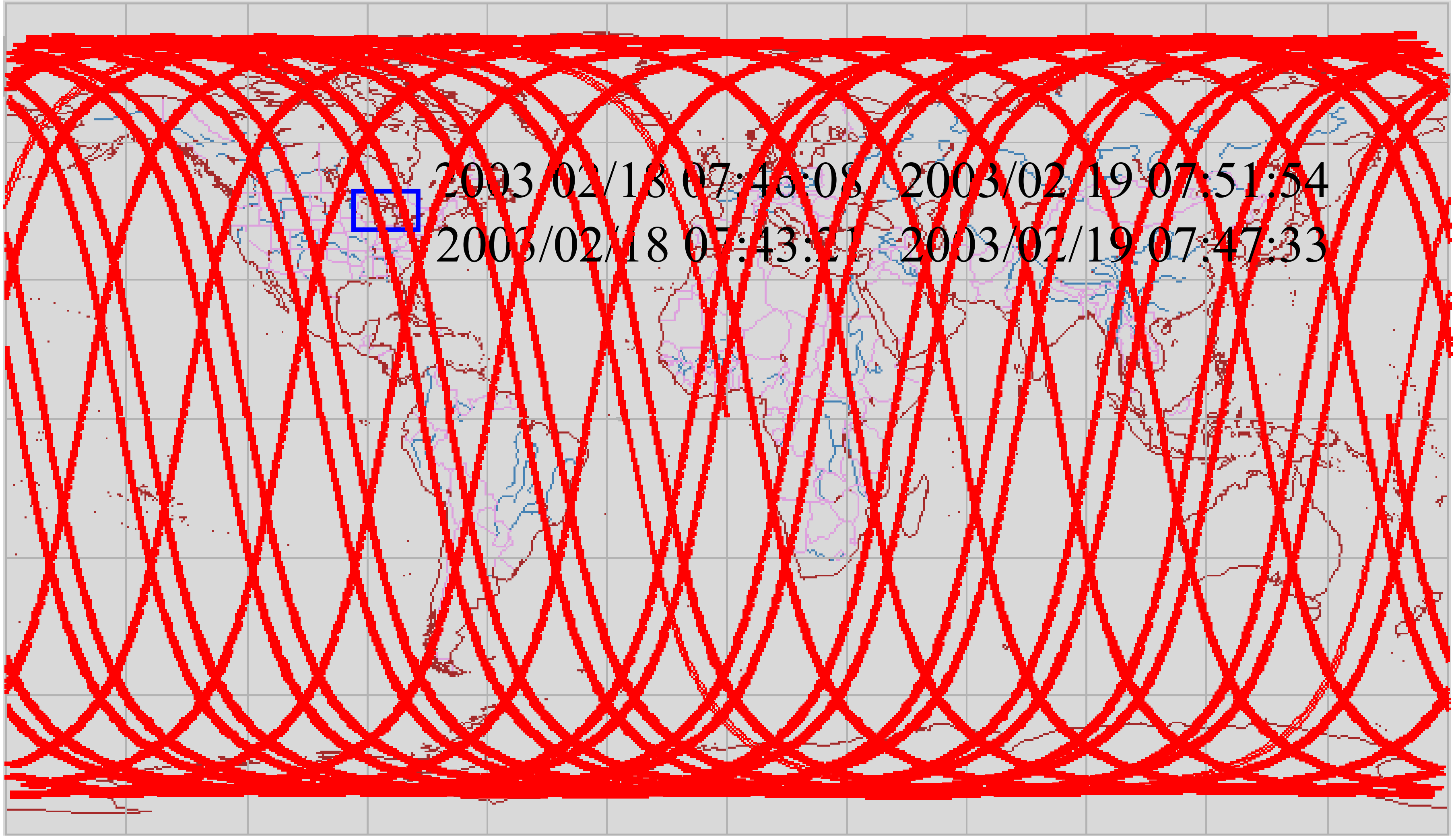
Worldwide Reference System



Nominal Orbit Spatial Extent



Predict – Orbit Propagator



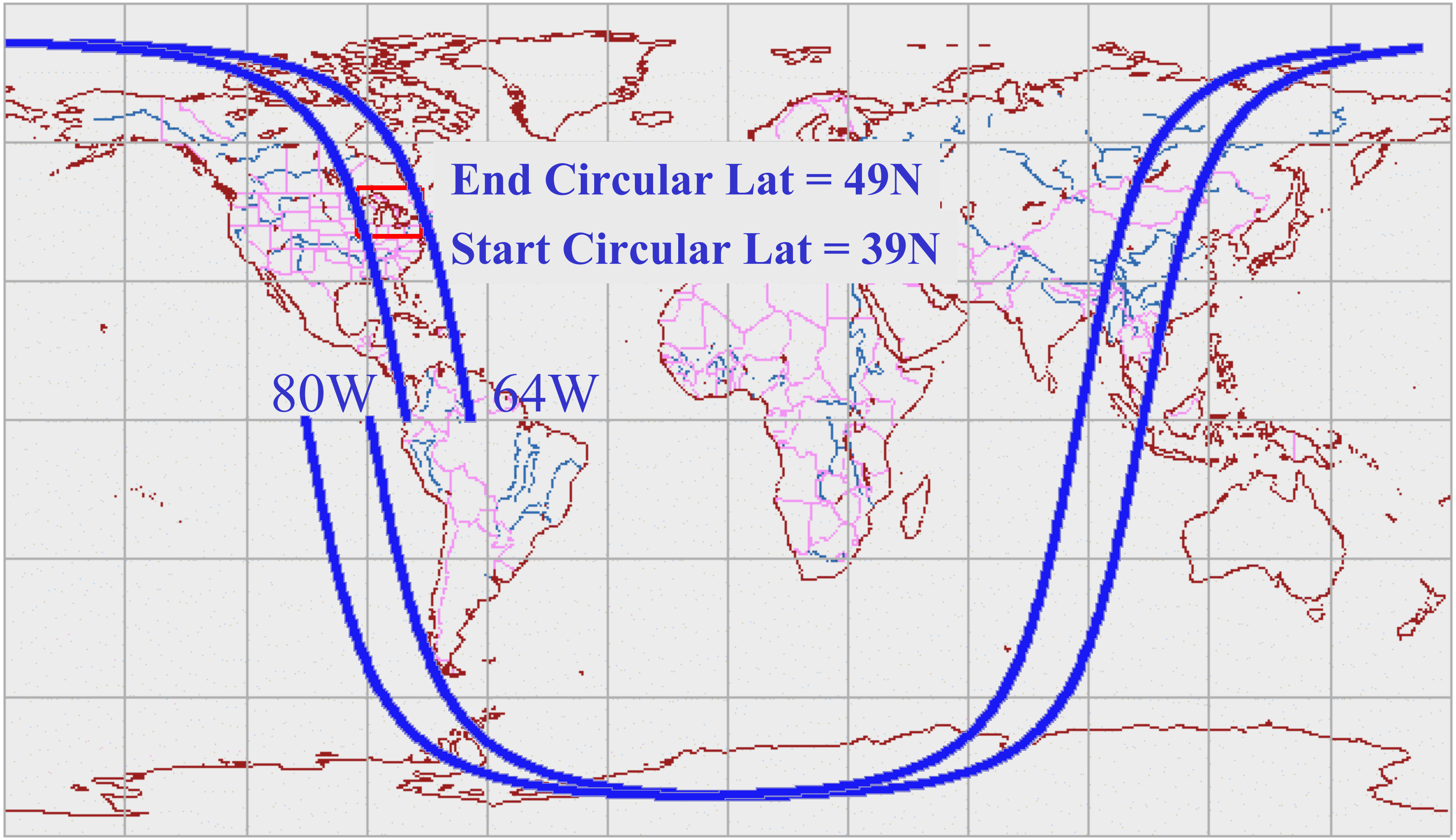
Predict SQL – Thule – one day

```
where (
('2003/02/18 01:23:03' between startDateTime and endDateTime) or
('2003/02/18 03:03:57' between startDateTime and endDateTime) or
('2003/02/18 04:44:34' between startDateTime and endDateTime) or
('2003/02/18 06:05:21' between startDateTime and endDateTime) or
('2003/02/18 07:46:32' between startDateTime and endDateTime) or
('2003/02/18 12:47:46' between startDateTime and endDateTime) or
('2003/02/18 14:10:32' between startDateTime and endDateTime) or
('2003/02/18 15:51:02' between startDateTime and endDateTime) or
('2003/02/18 17:11:31' between startDateTime and endDateTime) or
('2003/02/18 18:52:07' between startDateTime and endDateTime)
)
```

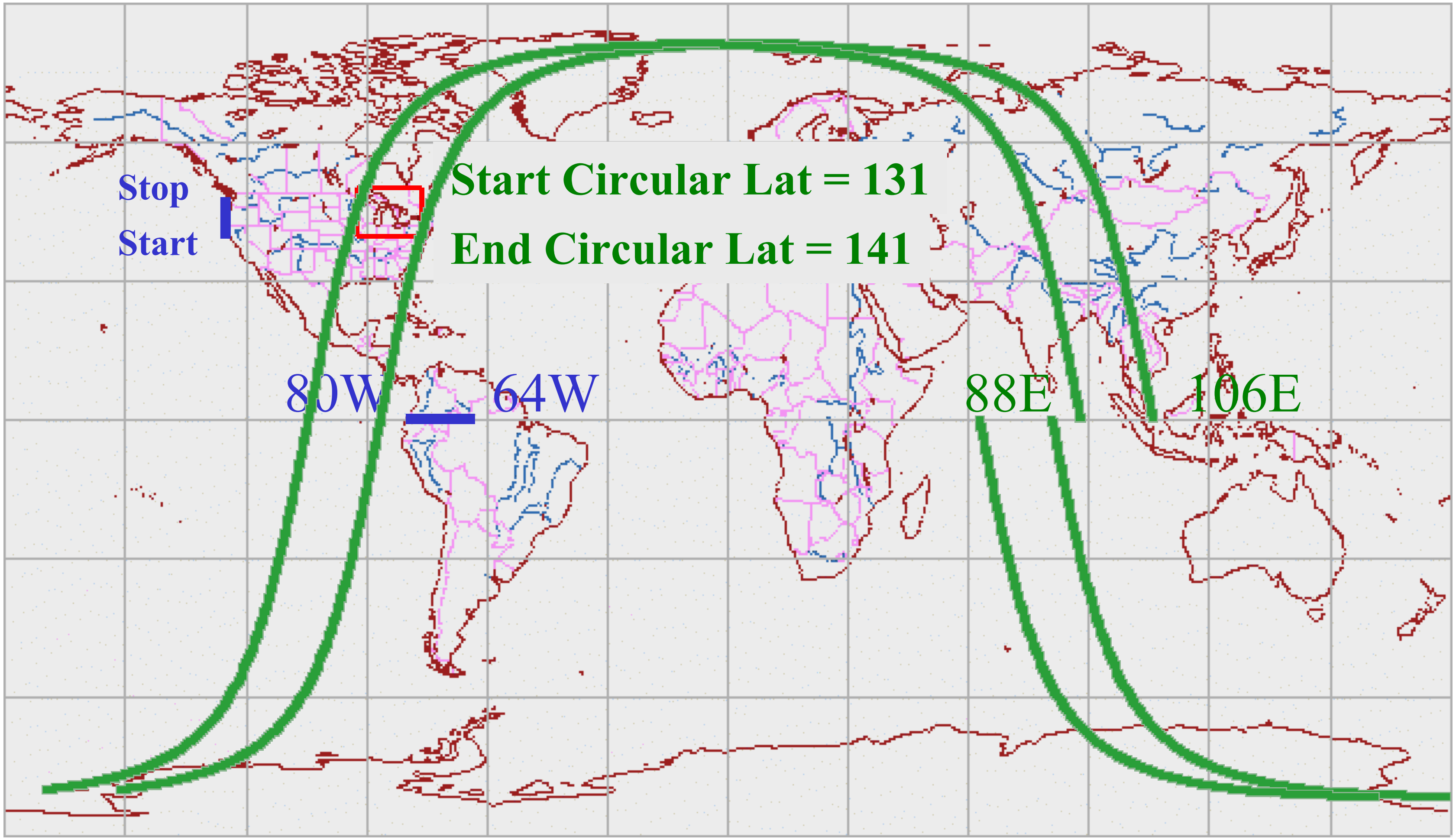
Predict SQL – Thule – Two Weeks

```
where (
('2003/02/18 01:23:03' between startDateTime and endDateTime) or ('2003/02/18 03:03:57' between startDateTime and endDateTime) or ('2003/02/18 04:44:34' between startDateTime and endDateTime) or
('2003/02/18 06:05:21' between startDateTime and endDateTime) or ('2003/02/18 07:46:32' between startDateTime and endDateTime) or ('2003/02/18 12:47:46' between startDateTime and endDateTime) or
('2003/02/18 14:10:32' between startDateTime and endDateTime) or ('2003/02/18 15:51:02' between startDateTime and endDateTime) or ('2003/02/18 17:11:31' between startDateTime and endDateTime) or
('2003/02/18 18:52:07' between startDateTime and endDateTime) or ('2003/02/19 01:23:03' between startDateTime and endDateTime) or ('2003/02/19 03:03:57' between startDateTime and endDateTime) or
('2003/02/19 04:44:34' between startDateTime and endDateTime) or ('2003/02/19 06:05:21' between startDateTime and endDateTime) or ('2003/02/19 07:46:32' between startDateTime and endDateTime) or
('2003/02/18 12:47:46' between startDateTime and endDateTime) or ('2003/02/19 14:10:32' between startDateTime and endDateTime) or ('2003/02/19 15:51:02' between startDateTime and endDateTime) or
('2003/02/18 17:11:31' between startDateTime and endDateTime) or ('2003/02/19 18:52:07' between startDateTime and endDateTime) or ('2003/02/20 01:23:03' between startDateTime and endDateTime) or
('2003/02/20 03:03:57' between startDateTime and endDateTime) or ('2003/02/20 04:44:34' between startDateTime and endDateTime) or ('2003/02/20 06:05:21' between startDateTime and endDateTime) or
('2003/02/20 07:46:32' between startDateTime and endDateTime) or ('2003/02/20 12:47:46' between startDateTime and endDateTime) or ('2003/02/20 14:10:32' between startDateTime and endDateTime) or
('2003/02/20 15:51:02' between startDateTime and endDateTime) or ('2003/02/20 17:11:31' between startDateTime and endDateTime) or ('2003/02/20 18:52:07' between startDateTime and endDateTime) or
('2003/02/21 01:23:03' between startDateTime and endDateTime) or ('2003/02/21 03:03:57' between startDateTime and endDateTime) or ('2003/02/21 04:44:34' between startDateTime and endDateTime) or
('2003/02/21 06:05:21' between startDateTime and endDateTime) or ('2003/02/21 07:46:32' between startDateTime and endDateTime) or ('2003/02/21 12:47:46' between startDateTime and endDateTime) or
('2003/02/21 14:10:32' between startDateTime and endDateTime) or ('2003/02/21 15:51:02' between startDateTime and endDateTime) or ('2003/02/21 17:11:31' between startDateTime and endDateTime) or
('2003/02/21 18:52:07' between startDateTime and endDateTime) or ('2003/02/22 01:23:03' between startDateTime and endDateTime) or ('2003/02/22 03:03:57' between startDateTime and endDateTime) or
('2003/02/22 04:44:34' between startDateTime and endDateTime) or ('2003/02/22 06:05:21' between startDateTime and endDateTime) or ('2003/02/22 07:46:32' between startDateTime and endDateTime) or
('2003/02/22 12:47:46' between startDateTime and endDateTime) or ('2003/02/22 14:10:32' between startDateTime and endDateTime) or ('2003/02/22 15:51:02' between startDateTime and endDateTime) or
('2003/02/22 17:11:31' between startDateTime and endDateTime) or ('2003/02/22 18:52:07' between startDateTime and endDateTime) or ('2003/02/23 01:23:03' between startDateTime and endDateTime) or
('2003/02/23 03:03:57' between startDateTime and endDateTime) or ('2003/02/23 04:44:34' between startDateTime and endDateTime) or ('2003/02/23 06:05:21' between startDateTime and endDateTime) or
('2003/02/23 07:46:32' between startDateTime and endDateTime) or ('2003/02/23 12:47:46' between startDateTime and endDateTime) or ('2003/02/23 14:10:32' between startDateTime and endDateTime) or
('2003/02/23 15:51:02' between startDateTime and endDateTime) or ('2003/02/23 17:11:31' between startDateTime and endDateTime) or ('2003/02/23 18:52:07' between startDateTime and endDateTime) or
('2003/02/24 01:23:03' between startDateTime and endDateTime) or ('2003/02/24 03:03:57' between startDateTime and endDateTime) or ('2003/02/24 04:44:34' between startDateTime and endDateTime) or
('2003/02/24 06:05:21' between startDateTime and endDateTime) or ('2003/02/24 07:46:32' between startDateTime and endDateTime) or ('2003/02/24 12:47:46' between startDateTime and endDateTime) or
('2003/02/24 14:10:32' between startDateTime and endDateTime) or ('2003/02/24 15:51:02' between startDateTime and endDateTime) or ('2003/02/24 17:11:31' between startDateTime and endDateTime) or
('2003/02/24 18:52:07' between startDateTime and endDateTime) or ('2003/02/25 01:23:03' between startDateTime and endDateTime) or ('2003/02/25 03:03:57' between startDateTime and endDateTime) or
('2003/02/25 04:44:34' between startDateTime and endDateTime) or ('2003/02/25 06:05:21' between startDateTime and endDateTime) or ('2003/02/25 07:46:32' between startDateTime and endDateTime) or
('2003/02/25 12:47:46' between startDateTime and endDateTime) or ('2003/02/25 14:10:32' between startDateTime and endDateTime) or ('2003/02/25 15:51:02' between startDateTime and endDateTime) or
('2003/02/25 17:11:31' between startDateTime and endDateTime) or ('2003/02/25 18:52:07' between startDateTime and endDateTime) or ('2003/02/26 01:23:03' between startDateTime and endDateTime) or
('2003/02/26 03:03:57' between startDateTime and endDateTime) or ('2003/02/26 04:44:34' between startDateTime and endDateTime) or ('2003/02/26 06:05:21' between startDateTime and endDateTime) or
('2003/02/26 07:46:32' between startDateTime and endDateTime) or ('2003/02/26 12:47:46' between startDateTime and endDateTime) or ('2003/02/26 14:10:32' between startDateTime and endDateTime) or
('2003/02/26 15:51:02' between startDateTime and endDateTime) or ('2003/02/26 17:11:31' between startDateTime and endDateTime) or ('2003/02/26 18:52:07' between startDateTime and endDateTime) or
('2003/02/27 01:23:03' between startDateTime and endDateTime) or ('2003/02/27 03:03:57' between startDateTime and endDateTime) or ('2003/02/27 04:44:34' between startDateTime and endDateTime) or
('2003/02/27 06:05:21' between startDateTime and endDateTime) or ('2003/02/27 07:46:32' between startDateTime and endDateTime) or ('2003/02/27 12:47:46' between startDateTime and endDateTime) or
('2003/02/27 14:10:32' between startDateTime and endDateTime) or ('2003/02/27 15:51:02' between startDateTime and endDateTime) or ('2003/02/27 17:11:31' between startDateTime and endDateTime) or
('2003/02/27 18:52:07' between startDateTime and endDateTime) or ('2003/02/28 01:23:03' between startDateTime and endDateTime) or ('2003/02/28 03:03:57' between startDateTime and endDateTime) or
('2003/02/28 04:44:34' between startDateTime and endDateTime) or ('2003/02/28 06:05:21' between startDateTime and endDateTime) or ('2003/02/28 07:46:32' between startDateTime and endDateTime) or
('2003/02/28 12:47:46' between startDateTime and endDateTime) or ('2003/02/28 14:10:32' between startDateTime and endDateTime) or ('2003/02/28 15:51:02' between startDateTime and endDateTime) or
('2003/02/28 17:11:31' between startDateTime and endDateTime) or ('2003/02/28 18:52:07' between startDateTime and endDateTime) or ('2003/03/01 01:23:03' between startDateTime and endDateTime) or
('2003/03/01 03:03:57' between startDateTime and endDateTime) or ('2003/03/01 04:44:34' between startDateTime and endDateTime) or ('2003/03/01 06:05:21' between startDateTime and endDateTime) or
('2003/03/01 07:46:32' between startDateTime and endDateTime) or ('2003/03/01 12:47:46' between startDateTime and endDateTime) or ('2003/03/01 14:10:32' between startDateTime and endDateTime) or
('2003/03/01 15:51:02' between startDateTime and endDateTime) or ('2003/03/01 17:11:31' between startDateTime and endDateTime) or ('2003/03/01 18:52:07' between startDateTime and endDateTime) or
('2003/03/02 01:23:03' between startDateTime and endDateTime) or ('2003/03/02 03:03:57' between startDateTime and endDateTime) or ('2003/03/02 04:44:34' between startDateTime and endDateTime) or
('2003/03/02 06:05:21' between startDateTime and endDateTime) or ('2003/03/02 07:46:32' between startDateTime and endDateTime) or ('2003/03/02 12:47:46' between startDateTime and endDateTime) or
('2003/03/02 14:10:32' between startDateTime and endDateTime) or ('2003/03/02 15:51:02' between startDateTime and endDateTime) or ('2003/03/02 17:11:31' between startDateTime and endDateTime) or
('2003/03/02 18:52:07' between startDateTime and endDateTime) or ('2003/03/03 01:23:03' between startDateTime and endDateTime) or ('2003/03/03 03:03:57' between startDateTime and endDateTime) or
('2003/03/03 04:44:34' between startDateTime and endDateTime) or ('2003/03/03 06:05:21' between startDateTime and endDateTime) or ('2003/03/03 07:46:32' between startDateTime and endDateTime) or
('2003/03/03 12:47:46' between startDateTime and endDateTime) or ('2003/03/03 14:10:32' between startDateTime and endDateTime) or ('2003/03/03 15:51:02' between startDateTime and endDateTime) or
('2003/03/03 17:11:31' between startDateTime and endDateTime) or ('2003/02/18 18:52:07' between startDateTime and endDateTime) )
```

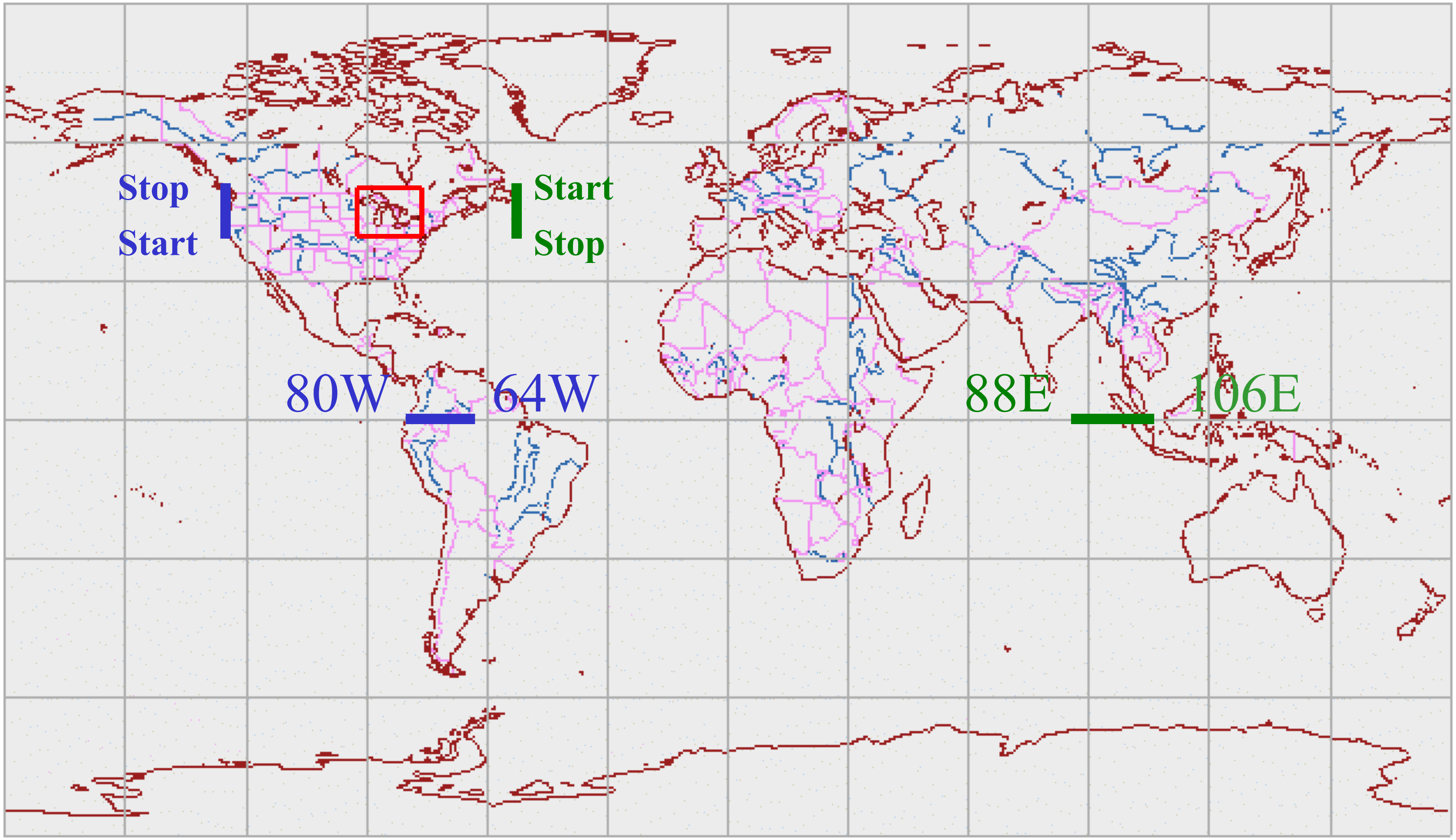
Backtrack (Ascending)



Bactrack (Descending)



Backtrack Parameters



Backtrack SQL

```
select distinct file_ID, start_date, data_set, ascending_crossing
from inventory where data_set like "AVHRR_LAC" and
start_date >= 19800101 and end_date <= 20000218 and
( ( ( (start_clat between 39.0 and 49.0) or
(end_clat between 39.0 and 49.0) or
(start_clat < 39.0 and end_clat > 49.0) ) and
( ascending_crossing between -80.0 and -64.0)  ) or
( (start_clat between 131.0 and 141.0) or
(end_clat between 131.0 and 141.0) or
(start_clat < 131.0 and end_clat > 141.0) ) and
(ascending_crossing between 88.0 and 106.0) ) ) )
```

References

Bates, Roger R., Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics*, Dover Publications, New York, 1971.

Heroux, David. "Leveraging Nominal Orbit Spatial Extent to Provide a Solution for the Archival of Geoscience Laser Altimeter System (GLAS) Products in ECS",, December, 2001.

Swick R. and K. Knowles, “Spatial Tools for a Round Planet”, Poster #OS51B-166 at the American Geophysical Union Fall Meeting, San Francisco, December 6-10, 2002.

Swick, R., and K. Knowles. "The Backtrack Orbit Search Algorithm", http://www.geospatialmethods.org/bosa, November, 2004.

Williams, Daryl. "The Worldwide Reference System (WRS)", http://landsat.gsfc.nasa.gov/documentation/wrs.html, May, 2004.

ECHO vs. ECS

Or

Backtrack vs. NOSE

ECHO using Backtrack:

Data: AMSR L2A
Period: 98.8 minutes
Inclination: 98.2 degrees
Swath Width: 1445 km.

ECS using NOSE

Data: AMSR L2A
Period 98.8 minutes
Inclination: 98.2 degrees
Swath Width: 1600 km.
Tracks: 720 half orbits
Blocks: 18 per track

Because NOSE uses nominal orbits the swath width has to be inflated so all the actual orbits represented by the nominal orbit fit. Using a NOSE implementation of one nominal orbit per degree means inflating the swath width by at least one degree, or 110 km. In this case they were actually inflated by 155 km.

Search:

Temporal: May 1, 2005 – May 25, 2005
Spatial: Lat {40, 65}, Lon {-80, -50}

Results:

ECHO: 130 granules
66 Ascending with crossings in {-77.64, -11.22}
64 Descending with crossings in {-118.57, -52.15}

ECS: 157 granules
80 Ascending with crossings in {-80.79, -5.04}
77 Descending with crossings in {-126.35, -49.07}

Analysis:

Given the wider swath width of the NOSE scheme we would expect NOSE to generate (155/1445) = 10.7% more results.

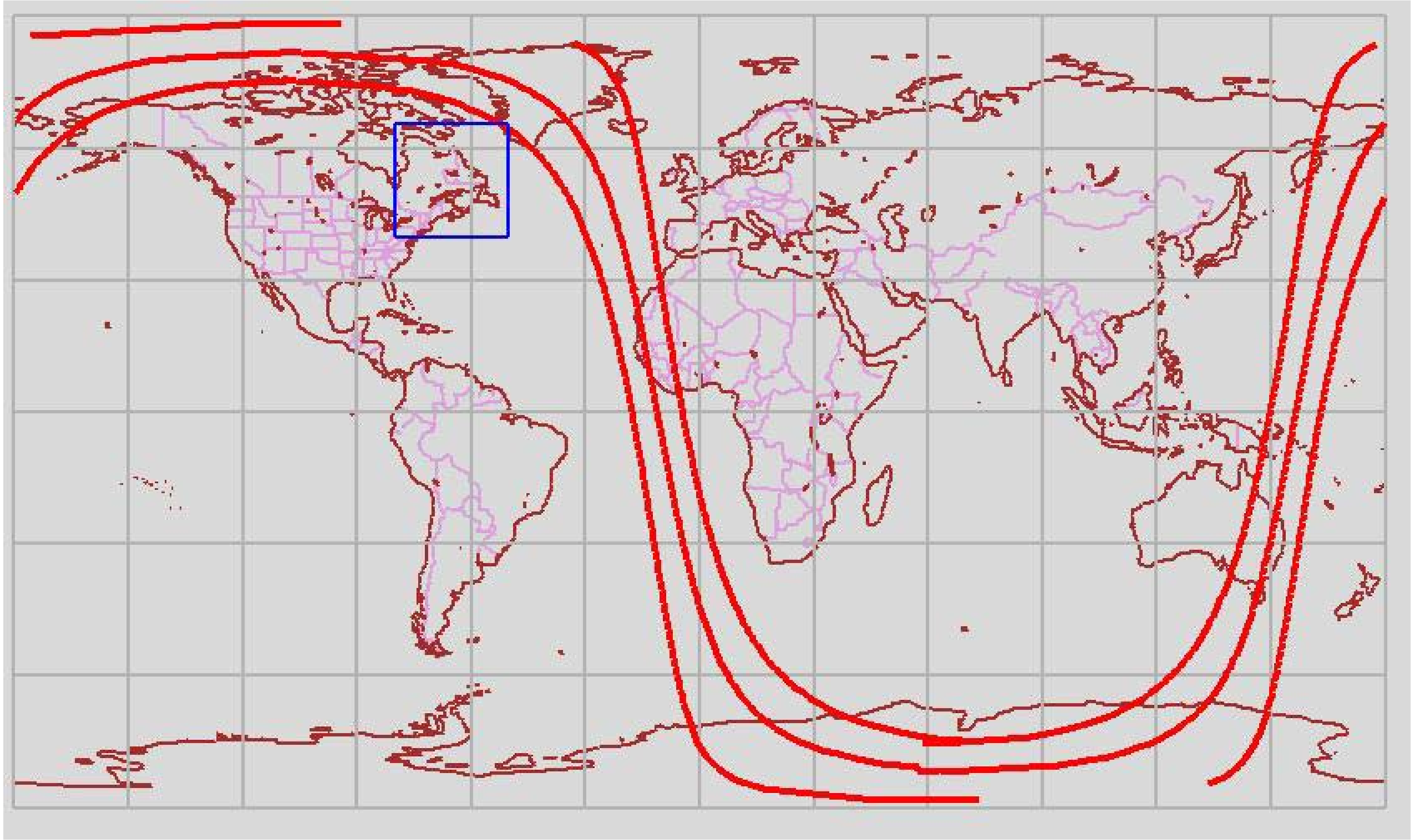
Instead we got (27/130) = 20.8% more results.

ECS/NOSE found 7 granules that ECHO/Backtrack “should” have found. This test was rather preliminary and when we checked ECHO we found the metadata for those 7 granules wasn’t in ECHO.

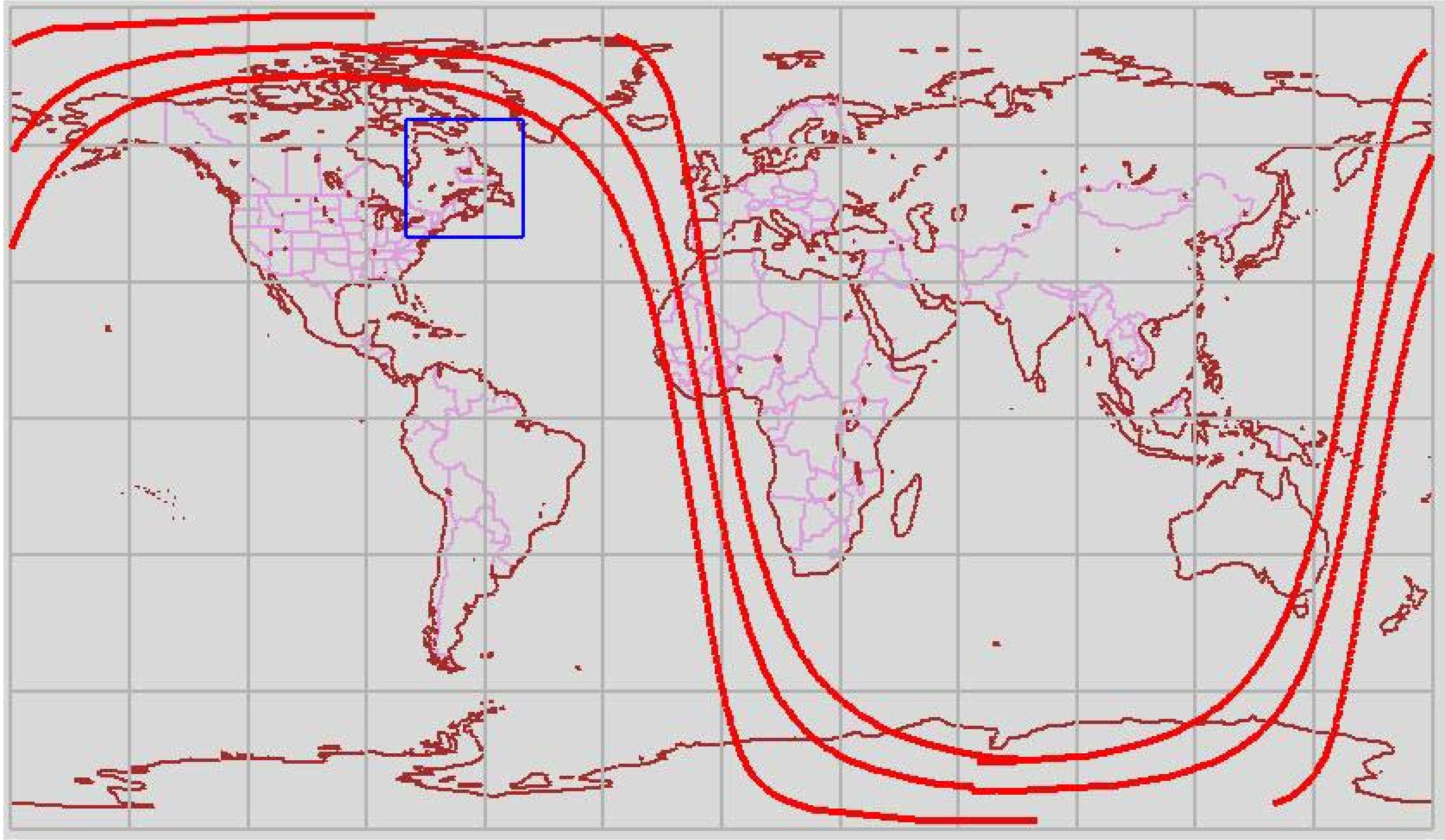
The other 20 “extra” granules had crossings outside the range returned by ECHO/Backtrack

A more controlled test can take place once ECHO 6.0 is operational

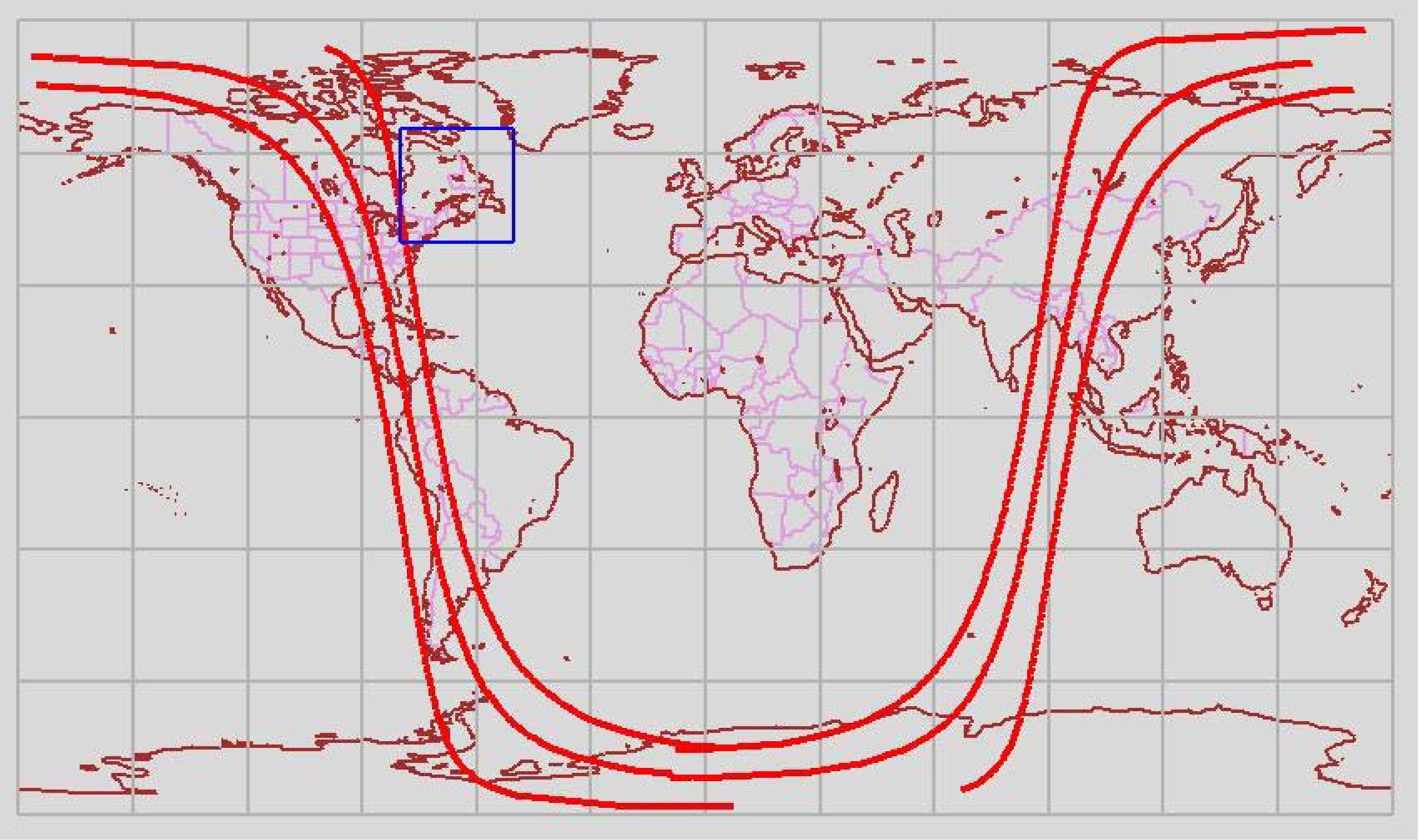
ECHO/Backtrack Maximum ascending crossing at -11.22 degrees



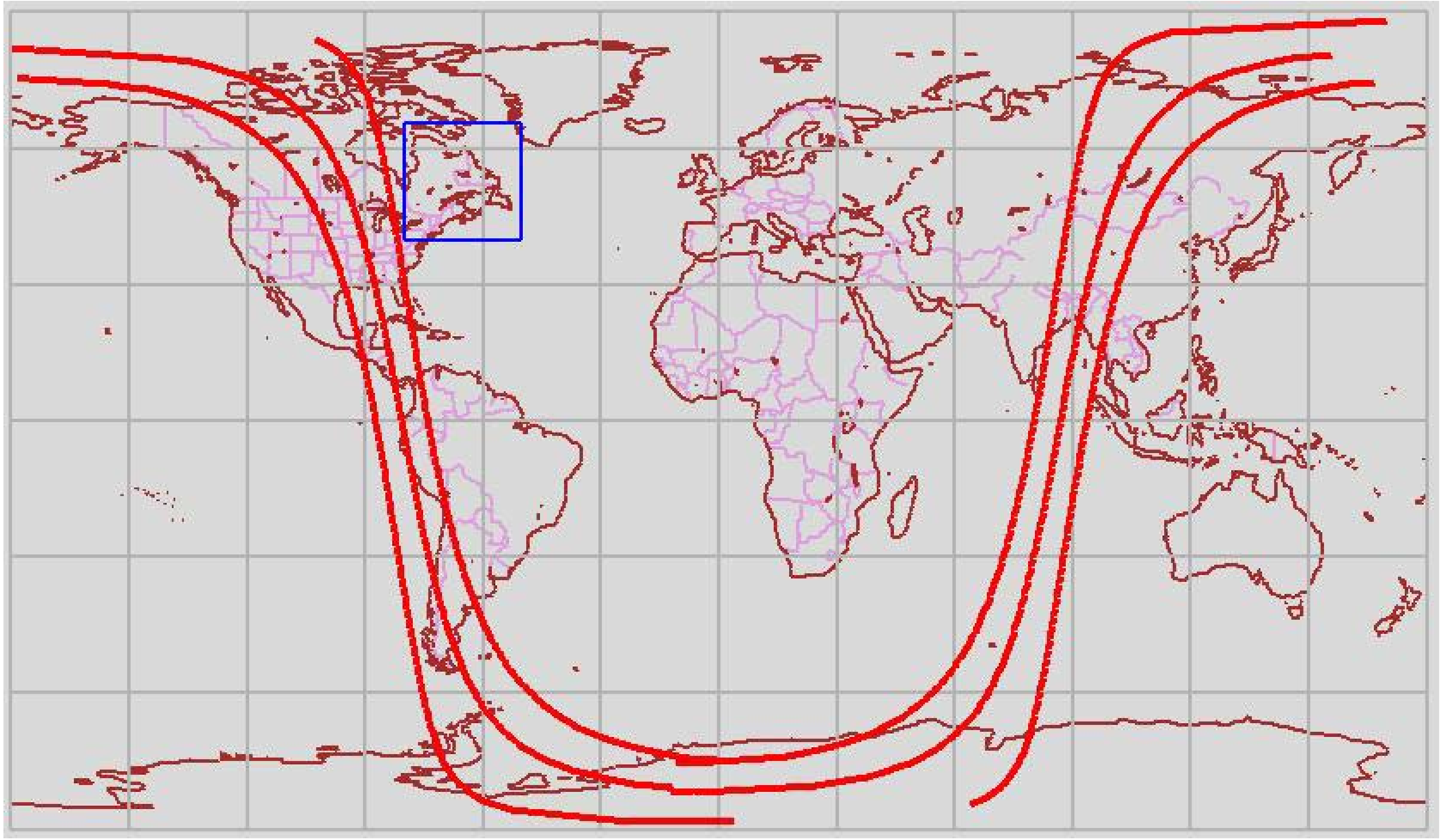
ECS/NOSE Maximum ascending crossing at -5.04 degrees



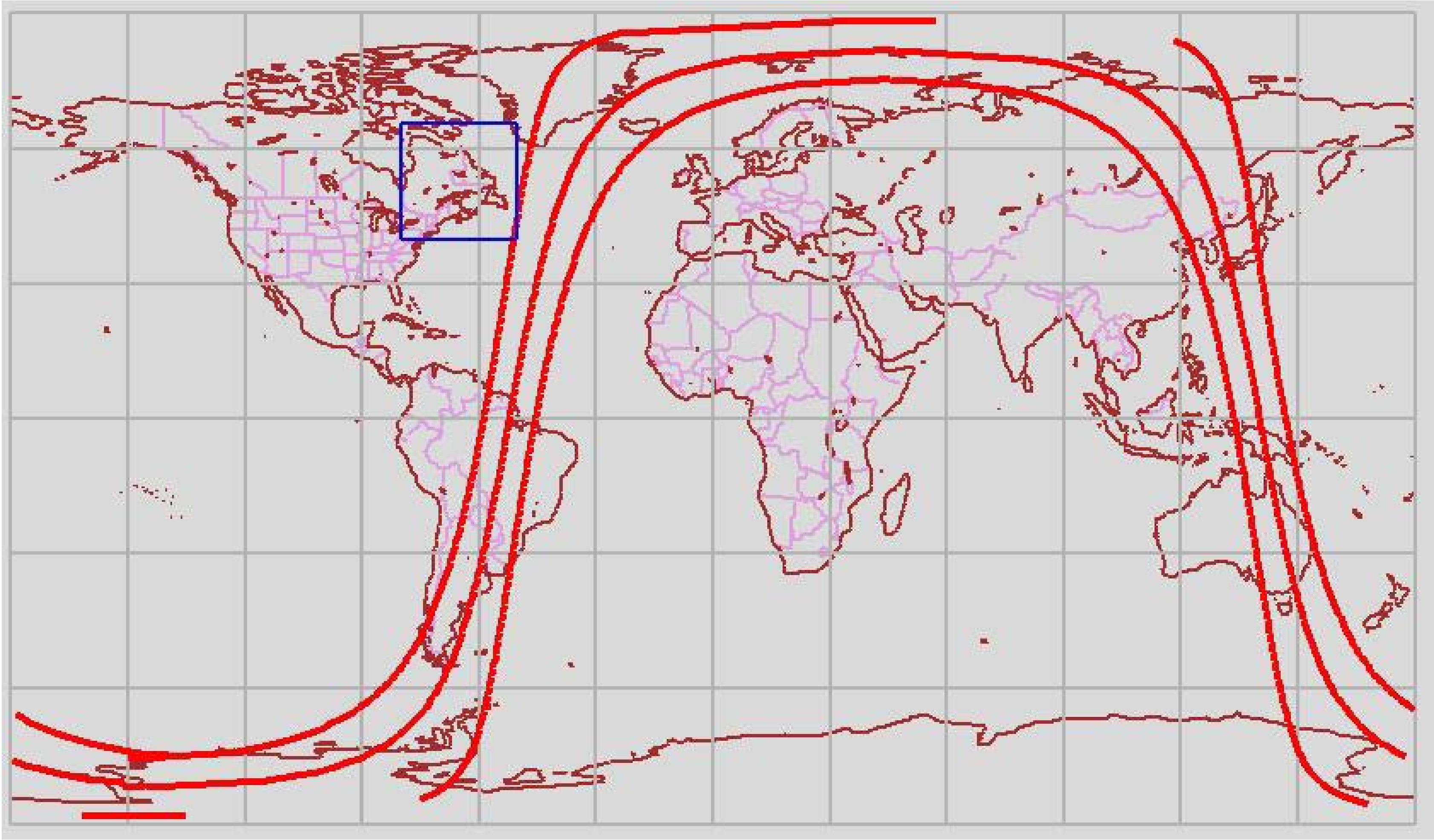
ECHO/Backtrack Minimum ascending crossing at -77.64 degrees



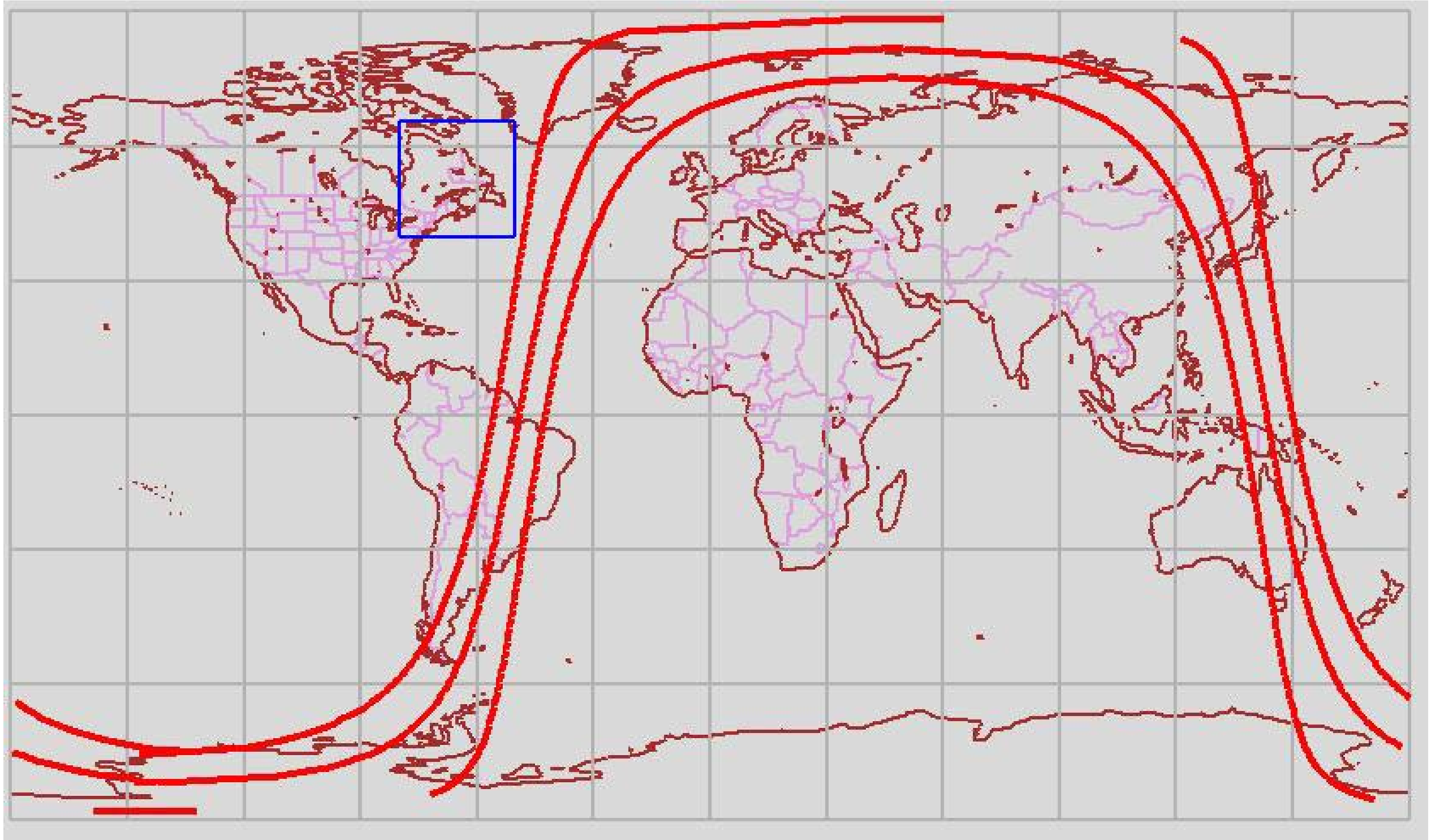
ECS/NOSE Minimum ascending crossing at –80.79 degrees



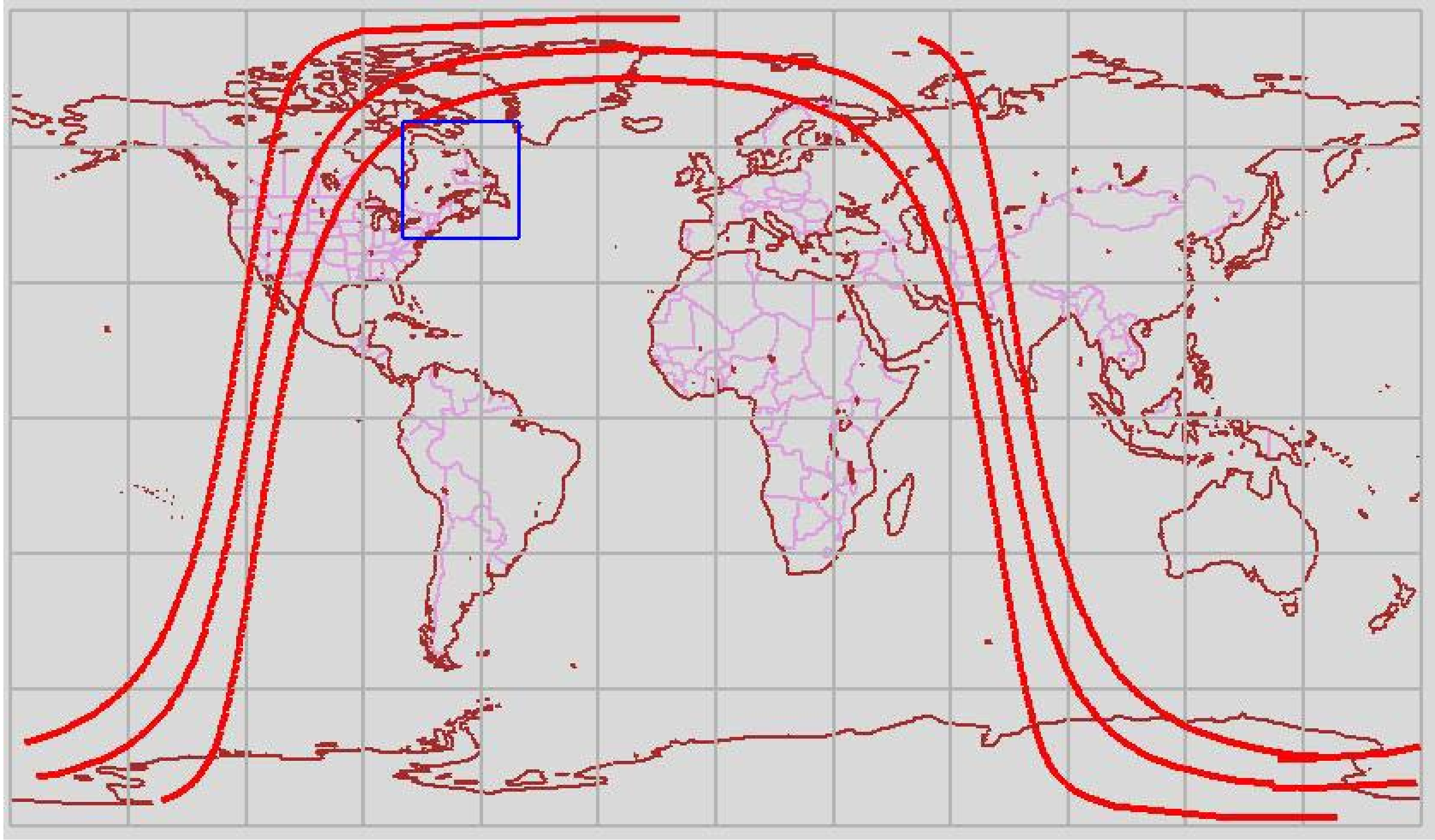
ECHO/Backtrack Maximum descending crossing at -52.15 degrees



ECS/NOSE Maximum descending crossing at -49.07 degrees



ECHO/Backtrack Minimum descending crossing at -118.57 degrees



ECS/NOSE Minimum descending crossing at -126.35 degrees